LA-UR-99-2394
*Approved for public release;
distribution is unlimited.*

Title:

Optimal Configuration of a Command and Control Network: Balancing Performance and Reconfiguration Constraints

Author(s): L. Jonathan Dowell

Submitted to:

Optimal Configuration of a Command and Control Network: Balancing Performance and Reconfiguration Constraints

L. Jonathan Dowell
Los Alamos National Laboratory
P. O. Box 1663 MS F604
Los Alamos, NM 87545
ljdowell@lanl.gov

ABSTRACT
The optimization of the configuration of communications and control networks is important for assuring the reliability and performance of the networks. This paper presents techniques for determining the optimal configuration for such a network in the presence of communication and connectivity constraints.

Keywords
network optimization, spanning tree, data-fusion network, genetic algorithm

1. INTRODUCTION
Many command and control systems, supervisory control and data acquisition systems, and data-fusion networks consist of a spanning tree that connects remote sensors to a central data-processing facility or command post. Such networks combine measurements from remote sensors at a single central processing facility. Examples of such networks include the supervisory control and data acquisition (SCADA) systems of electric-power and natural-gas utilities, and some military command and control systems.

Optimization of such networks is important for assuring the reliability and performance of the networks. Such networks can be characterized as graphs of vertices and edges, where the edges are the communications links between vertices of computer, communication, data-acquisition, and control devices. The networks contain a central processing facility, which can be a command post, control center, or other data-collection facility. Several types of constraints can be considered in the identification of an optimal configuration for the network. Such constraints include minimizing the probability of data loss or corruption from poor communications edges, reducing the number of edges between the central command post and the most distant vertices in the network, and allocating the connectivity between vertices while reserving connectivity capacity for reconfiguration of the network in the event of vertex or edge failures.

This paper examines the optimization of spanning-tree communications networks containing a single central processing facility. It presents an optimal configuration of a network for the minimization of the probability of data loss in communications links. It presents a genetic algorithm method for optimally embedding a spanning tree in a graph for the selection of a data-fusion network subgraph. Finally, it explores a technique for reconfiguration to restore connectivity to a data-fusion network following the failure of a network component.

LANL LA-UR 99-2394

ACM Symposium on Applied Computing
Villa Olmo, Como, Italy 19-21 Mar 2000
Paper ID #CS-03
accepted for publication 1 Nov 1999

2. OPTIMAL CONFIGURATION TO MINIMIZE DATA LOSS
A communications or control network can be described as a graph. The edges of this graph route messages from the remote vertices to a central processing facility (such as a command post), with relaying through other vertices as necessary. Consider this infrastructure network to be a strict, connected, undirected graph $G = (V, E)$ of vertices V and edges E. The command-post vertex is $\phi \in V$, and the other vertices $\{v\colon v \in V, v \neq \phi\}$ are the set of remote terminals and relaying stations. Associate with each edge $e \in E$ a positive weight $l(e)$ called the length of e. The spanning-tree subgraph $T = (V, E_T)$, $E_T \subseteq E$, is a connected subgraph of G without cycles. The edges E_T consist of the active edges through which messages are routed from the remote vertices to the command post. That is, there are a variety of edges connecting the various vertices of G, but only some of the edges are used in the data-fusion network. The edges of E_T are selected from E to optimize T by some metric. In a real-world network, the optimization may be to maximize the signal-to-noise ratio of the resulting paths, reduce the number of edges between the command post and the remote terminals in the longest paths, minimize the probability of data loss, or produce an extreme value for some other practical metric. The length a_1a_2 of a path between vertices a_1 and a_2 is the sum of the lengths of the edges belonging to the unique path $P_{a_1a_2}$ in T between a_1 and a_2:

$$a_1a_2 = \text{length of path } P_{a_1a_2} = \sum_{e \in P_{a_1a_2}} l(e) \tag{1}$$

The radius $r(v)$ of a vertex v is the distance $v\phi$ between v and the command-post vertex ϕ. The degree or valency of a vertex is the number of edges incident with that vertex.

These concepts can be used to optimize the configuration of a spanning-tree communications network to minimize the probability of data loss in the network's communications. Let $\rho(e)$ be the independent probability of successfully transmitting a message through edge e. Then the probability $Pr(v)$ of successfully transmitting a message from vertex v to the command post ϕ is the product of the probabilities of successful transmission of each of the edges in the path P between v and ϕ:

$$Pr(v) = \prod_{e \in P} \rho(e) \tag{2}$$

Then

$$Pr(v) = \exp\left(\ln\left(\prod_{e\in P}\rho(e)\right)\right) = \exp\left(-\sum_{e\in P}\ln\left(\tfrac{1}{\rho(e)}\right)\right) \tag{3}$$

Note that $Pr(v)$ is maximized when $\sum_{e\in P}\ln\left(\frac{1}{\rho(e)}\right)$ is minimized. Then the optimal configuration of this spanning-tree communications network is a shortest-path tree subgraph where the length function $l(e)$ is the natural logarithm of the reciprocal of $\rho(e)$, $l(e)=\ln\left(\frac{1}{\rho(e)}\right)$. The shortest-path tree subgraph can be found with Dijkstra's famous algorithm [1,2].

3. OPTIMAL CONFIGURATION WITH CONNECTIVITY CONSTRAINTS

The method described in section 2 supposes that any edge $e\in E$ that exists can be employed in the fabrication of the spanning-tree communications networks. However, there may be practical constraints to the connectivity that is permitted in the realization of this network. For instance, there may be an upper bound to the degree for each vertex, there may be a maximum number of vertices permitted in any branch of the tree, the graph *G* may be incomplete, or other similar constraints may exist.

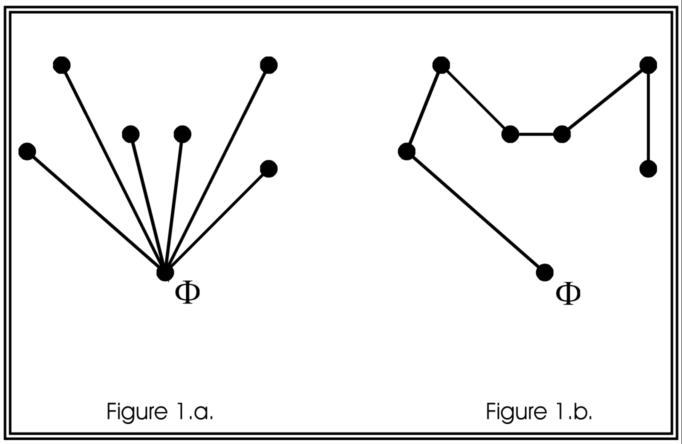

Figure 1. Connectivity constraints produce different spanning-tree communication networks. Figure 1a shows a star graph, minimizing the distance of each vertex from the command post ϕ, but demanding a maximum degree for the command-post vertex. Figure 1b shows a chain graph, minimizing the degree of each vertex but maximizing the distances of vertices from the command post.

Figure 1 illustrates two extreme examples of possible configurations of a communications network. Figure 1a shows a star graph. In this graph, every vertex is connected directly to the command post. The number of vertices in each branch is minimized, but the degree of the command-post vertex is maximized. Figure 1b shows a chain graph. The degree of the command post and of the other vertices in the graph is minimized, but the number of vertices in the branch is maximized.

Suppose the graph *G* is complete, the edge length is the same for all edges (that is, $l(e)=l\ \forall\ e\in E$), and there is an upper bound *k* for the degree for each vertex in the spanning tree subgraph *T*. Let us define optimality for a communications-network subgraph as the subgraph that minimizes the sum of the radii of each vertex over all vertices $v\in E$. Then the optimal communications-network subgraph is a fractal graph as shown in Fig. 2. *Proof*: This can be shown to be true by induction. As illustrated in Fig. 2, the fractal graph sorts the vertices *V* into layers as measured by the number of edges between a vertex $v_i\in$ layer *i* and ϕ. As the length $l(e)$ is uniform for all edges, the radius of a vertex in layer *i* is $(i\text{-}1)\cdot l$. For the first vertex $v\neq\phi$, the only possible location is in layer 2, with one edge between this first vertex and the command post ϕ. The location of minimal radius for the first *k* vertices is in layer 2. Then the degree of the command post has attained its maximum value, and subsequent vertices must be placed in higher-order layers. For subsequent vertices, say, the j^{th} vertex, the location of minimum radius is in the last unfilled layer of order *z*, where $j\leq\sum_{\alpha=2}^{z}k(k-1)^{\alpha-2}$ and *z* is the smallest integer satisfying this relation. Because the graph is complete, the edges required to make this location assignment (placing the j^{th} vertex in the z^{th} layer) are always available. As the radius of each vertex is minimized at the time of its assignment, the sum of the radii over all vertices is minimized, QED. Note that this optimal fractal graph is produced by a breadth-first search constrained by the maximum degree for each node.

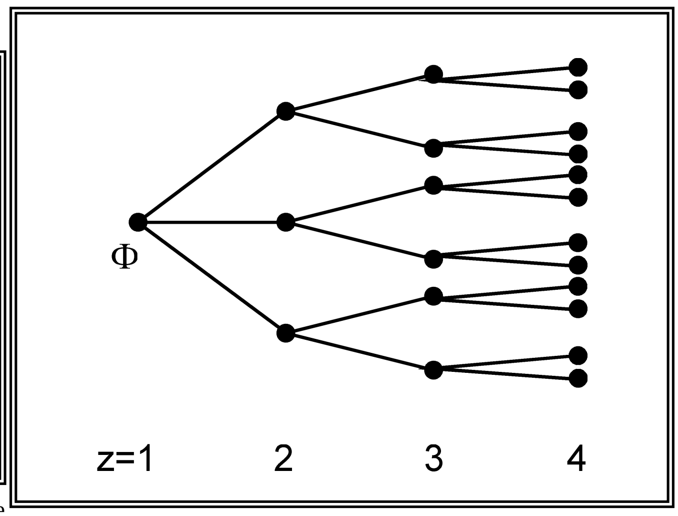

Figure 2. A fractal tree graph results from limiting the degree of the vertices in the graph while minimizing the sum of the distances of the vertices from the command post ϕ. Such a fractal graph is a subgraph of a complete graph, but may not be possible as a subgraph of incomplete but practical communication-network graphs.

If the graph G is not complete, then the edges necessary for the placement of the vertices in this optimal fractal arrangement may not be available. The fractal network structure is still optimal if the necessary edges are available. However, if there are vertices with degree smaller than the communication-network connectivity upper bound *k*, then these vertices present an inherent limit to the optimality of the realized network. The objective becomes to place such vertices in a layer with the highest possible order, which will minimize the consequence to the optimality of the network.

The problem of ordering the vertices to place those of deficient degree into a high-order layer is *NP*-complete. A method for a solution of this problem is a genetic algorithm. Genetic algorithms have been used recently to obtain efficient, practical solutions to a number of *NP*-complete graph problems [4,5,9]. Figures 3, 4, and 5 present an example of optimally embedding a degree-constrained communication-network subgraph in a graph of communications links.

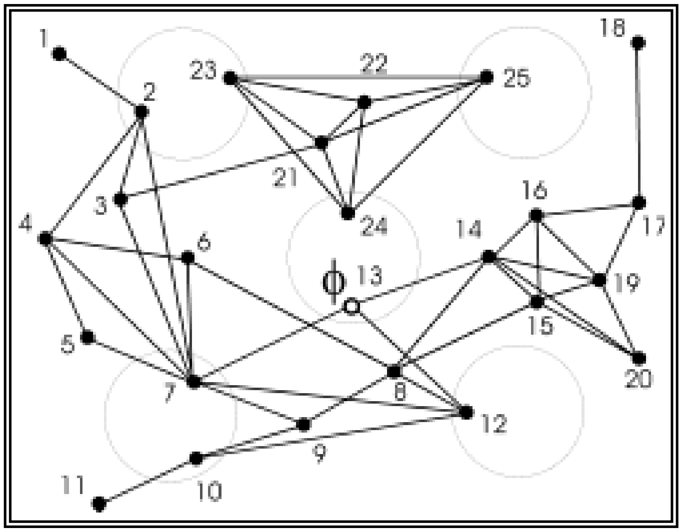

Figure 3. An example of a 25-vertex communication network. Communication links are limited by hills (represented by the large circles) and other obstacles, limiting this graph to 46 edges. The command post vertex ϕ is shown at vertex 13.

Figure 3 represents a communications network showing 25 remote terminals and relay stations deployed across terrain with hills and other obstacles that limit communications between the vertices. The central processing facility is a command post at node 13. The goal of this example is to select a communication-network tree subgraph subject to a constraint that the maximum degree for any vertex is 3, and with the objective of minimizing the number of edges between each vertex and the command post. A genetic algorithm is used to search the space of subgraphs, selecting a subgraph with 24 edges. With 46 edges in the communications network, there are over 7.89×10^{12} possible 24-edged subgraphs.

A chromosomal encoding scheme is necessary for the genetic algorithm. For this problem, the chromosomal encoding of the subgraph information is a record of the single adjacent vertex associated with each vertex of the graph. That is, when a spanning tree is found there will be a single path $v\phi$ between vertex v and the command post ϕ, for each $v\in V$. Then a sufficient chromosomal encoding records the unique vertex v' that is adjacent to v in this path. The genetic algorithm begins with a randomized population of 200 individual 24-edged subgraphs, where each adjacent vertex v' is selected at random from the vertices adjacent to v. The command-post vertex is omitted from this chromosome, as there will be no path needed to connect it to itself in the communication-network tree subgraph. Note that these initial individual chromosomes may not be complete trees, and may contain detached loops. An obvious fitness function for the genetic algorithm is the optimization metric for the tree subgraph, which is the sum of the layer numbers of each of the vertices in the tree. (The objective is to minimize the layer number for each vertex, minimizing the radius of each vertex, as was described previously in this section.) Note that the maximum possible layer number for a vertex connected to the command post by a 24-edged tree is 25. Any detached vertices are assigned a value of 100. Then the fitness function produces a value between the minimum of 49 (for an unconstrained 24-edged tree containing 24 vertices in layer 2) and the maximum of 2401 (for a subgraph with 24 vertices detached from the command post). With the constraint of a maximum degree of 3 for any vertex, the fitness function attains a minimum value of 88 for the optimal fractal subgraph having 25 vertices.

Genetic recombination creates a chromosome for a new individual by selecting an adjacent vertex v' for each vertex $v\in V$. This selection is made randomly for each vertex from the two chromosomes of the two parent individuals selected from the population. For each new chromosome, two parent chromosomes are selected by weighting the individuals in the population by the values of their fitness functions. Because the fitness function is a value to be minimized, individuals with lower fitness functions are more suitable and are favored in the selection of parents. For this experiment, a parent was selected by using the cube of the reciprocal of its fitness value as a weight. Once two parents were selected, then the new chromosome was developed by randomly selecting the adjacent vertex v' for each vertex v from one of the two parents. For this experiment, the first parent's gene was used 40% of the time, the second parent's gene was used 40% of the time, and 20% of the time mutation was introduced by randomly selecting an adjacent vertex v' from the adjacency list of vertex v. The constraint of a maximum degree of 3 for every vertex was imposed by discarding any new chromosomes that violated this constraint. Selection continued in each iteration until 200 new viable chromosomes were produced.

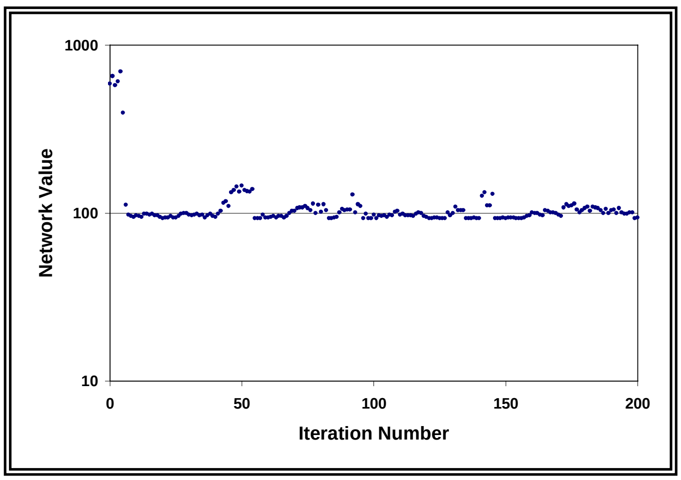

Figure 4. Selection of a spanning-tree subgraph for a data-fusion network with maximum degree of 3 is accomplished by a genetic algorithm (GA). The GA combines chromosomes of 24-edged subgraphs to find an optimal spanning tree of minimum total vertex radii. After 7 iterations (examining 1400 individuals), the GA produces a subgraph of value 98, and after 20 iterations produces a subgraph of value 93. An optimal fractal subgraph of 25 vertices has a value of 88.

Figure 4 illustrates the success of this genetic-algorithm experiment. An optimal fractal subgraph cannot be embedded in the graph in this example, but a nearly optimal fitness value of 93 was obtained after 20 iteration of genetic recombination (representing examination of only 4000 individuals from the 24-edged subgraph sample space). A nearly optimal fitness value of 98 was attained after 7 iterations, representing examination of 1400 individual chromosomes. In 200 iterations, six distinct subgraphs with fitness value of 93 were identified. Figure 5 shows an example of an optimal spanning-tree subgraph having fitness value of 93. The chromosome for this subgraph is:

Node	1	2	3	4	5	6	7	8	9	10
Adjacent	2	3	7	7	4	8	13	12	8	12
Node	11	12	13	14	15	16	17	18	19	20
Adjacent	10	13	0	13	14	14	16	17	15	15
Node	21	22	23	24	25					
Adjacent	3	21	21	23	22					

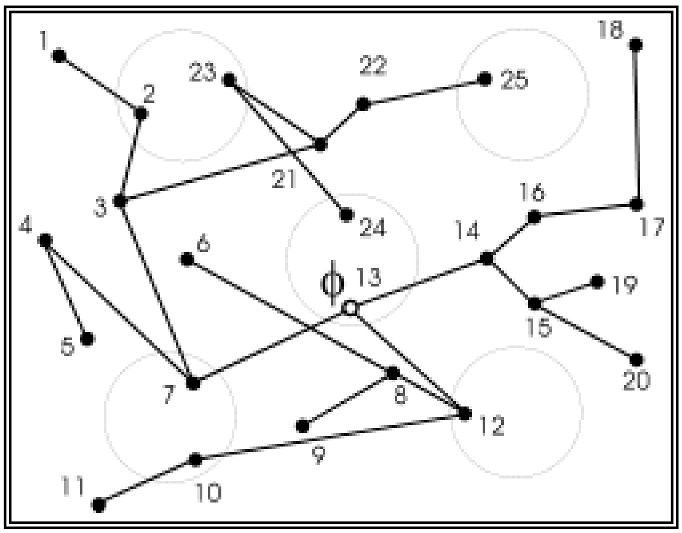

Figure 5. An example of a spanning-tree subgraph with maximum degree of 3 and total value of 93.

4. RECONFIGURATION

Present problems in assuring infrastructure security include optimizing the restoration of the infrastructure following the failure of one or more network elements. Let us consider failures of elements of a data-fusion network that cannot be repaired and that will require reconfiguration. Reconfiguration of two detached components consists of adding a new edge with one endpoint in each component so that communication between the command post and the disconnected component is restored. The set of edges $E_T^c = \{e \in E, e \notin E_T\}$ is the complement of E_T and is the set of redundant or unused edges available for reconfiguration of the network following a failure of a vertex or edge of the tree. A failure of an edge $e \in E_T$ will partition the tree into two detached components. An edge failure will require the addition of one new edge to reconnect the two detached components. A failure of a vertex will remove the vertex from the graph and will cause failures of the edges terminating at the failed vertex. If the vertex is a leaf, then the remaining graph will have a single component, and no reconfiguration will be required. If the vertex is not a leaf, then the failure will partition the graph into λ components, where λ is the degree of the failed vertex. Then λ–1 edges must be added to the graph to reconfigure these components. Because the operation of the network depends on operation of the command-post vertex, a failure of the command post cannot be reconfigured. Figure 6 illustrates reconfiguration following an edge failure.

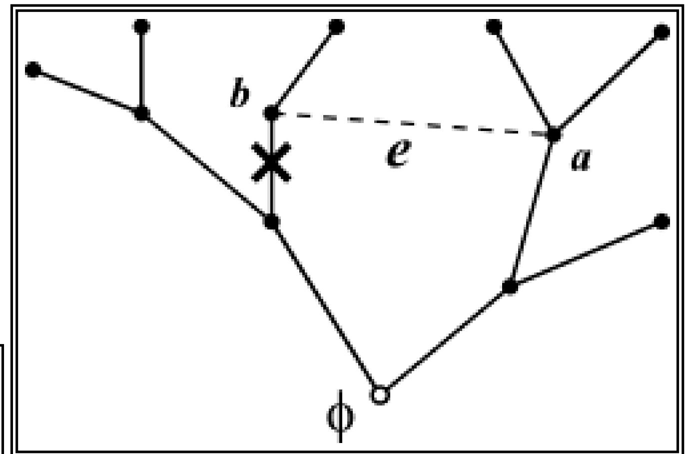

Figure 6. An example of reconfiguration of a data-fusion network following the failure of an edge of *T*. The cross indicates the failed edge. A new edge *e* is added such that one endpoint of *e* is in each of the two detached components of *T*. Endpoint *a* is in the component containing the command-post vertex ϕ, and endpoint *b* is in the component disconnected by the edge failure.

Reconfiguration of two detached components requires the addition of a new edge having an endpoint in each component. Let this edge e_{ab} have endpoint vertex *a* in the component containing the command post and endpoint vertex *b* in the detached component. Then the maximum radius of all vertices in the detached component following reconfiguration is $r(d) = r(a) + l(e_{ab}) + bd$, where *d* is the vertex in the detached component that is farthest from *b*. If there are *n* vertices in the detached component, then the distance *bd* can be determined in $O(n)$ by traversing the detached component. If there are *m* edges having one endpoint in the component containing the command post and the other endpoint in the detached component, then the optimal reconfiguration that will reduce the maximum radius $r(d)$ can be determined in $O(mn)$.

The computation time for optimal reconfiguration of a detached component can be improved by applying results developed by Handler. In [3], Handler presented several important results related to the vertex one center [6,7] of a tree. First, Handler describes an $O(n)$ algorithm for identifying the absolute one center and vertex one center of a tree. Second, Handler showed that a vertex one center is an element of every longest path $p(L(v))$ between a vertex *v* and the leaf $L(v)$ of greatest distance from *v*. Third, Handler described the diameter of *T* of which the vertex one center must be an element. Note that Handler's results reveal that a tree has a maximal branch relative to the vertex one center. This maximal branch contains *c*, the leaf farthest from the vertex one center. The tree also has a secondary, submaximal branch containing leaf c', the leaf of a distinct branch that is the second farthest from the vertex one center. Furthermore, the distance cc' is the diameter of *T*.

The vertex one center of the detached component, the distance $R(v)$ of every vertex *v* in the detached component from the vertex one center, and the membership of a vertex *v* in the maximal branch containing c can be found in $O(n)$. Then the maximum distance $vL(v)$ for every vertex v can be determined subsequently in $O(1)$.

$$vL(v) = \begin{cases} R(v) + R(c'), v \in \text{branch containing } c \\ R(v) + R(c),\ v \notin \text{branch containing } c \end{cases}. \tag{4}$$

Optimal reconfiguration is to find a new edge ab such that

$$r(d) = r(a) + l(e_{ab}) + bd \tag{5}$$

is minimized. Applying Handler's results,

$$bd = \begin{cases} R(b) + R(c'), b \in \text{branch containing } c \\ R(b) + R(c),\ b \notin \text{branch containing } c \end{cases}. \tag{6}$$

Then optimal reconfiguration can be determined in $O(m+n)$, which is more efficient for most m and n.

This implementation of Handler's algorithm for the purpose of reconfiguration following a network disconnection suggests an optimization metric for the selection of the communication-network tree subgraph, by selecting the tree so that edges between the branches (relative to the central processing facility) are left available for reconfiguration. Of course, this leads to a system of multiple objectives, with some objectives optimizing communication and reliability of the initial communication-network tree, and other objectives optimizing unused edges to be available for restoration. Conflicts between these objectives are likely to produce compromises between performance and reconfigurabilty, may be resolved through multi-objective genetic algorithms [8], and present a topic for further research in this area.

5. ACKNOWLEDGMENTS

Los Alamos National Laboratory (LANL) supported this work through the ELISIMS project and through work for the Joint Program Office for Biological Defense of the United States Department of Defense. The University of California operates LANL for the United States Department of Energy under contract *W-7405-ENG-36*. LANL strongly supports academic freedom and a researcher's right to publish; therefore, LANL as an institution does not endorse the viewpoint of a publication or guarantee its technical correctness.

6. REFERENCES

[1] E. W. Dijkstra, A Note on Two Problems in Connexion with Graphs, Numerische Mathematik 1 (1959) 269-271.

[2] E. Minieka, Optimization algorithms for networks and graphs. M. Dekker, New York, 1978.

[3] G. Y. Handler, Minimax Location of a Facility in an Undirected Tree Graph, Transportation Science 7 (1973) 287–293.

[4] B. A. Julstrom, Coding TSP Tours as Permutations via an Insertion Heuristic. Proceedings of the 1999 ACM Symposium on Applied Computing (San Antonio TX, Feb.-Mar. 1999), ACM, 297-301.

[5] K. Katayama and H. Narihisa, A New Iterated Local Search Algorithm using Genetic Crossover for the Traveling Salesman Problem. Proceedings of the 1999 ACM Symposium on Applied Computing (San Antonio TX, Feb.-Mar. 1999), ACM, 302-306.

[6] B. C. Tansel, R. L. Francis, and T. J. Lowe, Location on Networks: A Survey. Part I. The *p*-Center and *p*-Median Problems, Management Science 29 (1983) 482–497.

[7] B. C. Tansel, R. L. Francis, and T. J. Lowe, Location on Networks: A Survey. Part II. Exploiting Tree Network Structure, Management Science 29 (1983) 498–511.

[8] D. A. Van Veldhuizen and G. B. Lamont, Multiobjective Evolutionary Algorithm Test Suites. Proceedings of the 1999 ACM Symposium on Applied Computing (San Antonio TX, Feb.-Mar. 1999), ACM, 351-357.

[9] D. Whitley, T. Starkweather, and D. Shaner, The Traveling Salesman and Sequence Scheduling: Quality Solutions Using Genetic Edge Recombination. In L. Davis, ed., Handbook of Genetic Algorithms. Van Nostrand Reinhold, New York, 1991.

7. ABOUT THE AUTHOR

Dr. L. Jonathan Dowell is a Technical Staff Member in the Energy and Environmental Analysis Group at LANL. Dr. Dowell received his Ph.D. in engineering physics from the University of Virginia in 1989. Dr. Dowell has performed research and analysis in the fields of metrology, power-systems engineering, computer science, and applied mathematics. Dr. Dowell also is president of ReefNews, Inc., a non-profit organization that produces educational materials about the oceans.